Reasoning Systems, Inc.

Reengineering and Rewriting Legacy Software Systems

One of the key issues in significantly changing how software is composed and used is what to do about the existing software. The installed systems and data files—the so-called "legacy systems"—represent far too great an investment to be discarded, regardless of how useful the new software technology may be. A major difficulty is the wide variety of legacy systems, which might be written in any of dozens of computer languages and language "dialects," as well as countless specialized data formats. In 1984, Reasoning Systems, Inc., was launched to develop programs to fix software problems related to complex legacy systems. During the late 1980s, Reasoning created a family of reusable components for reengineering in the COBOL, C, FORTRAN, and Ada programming languages. These components were designed for the customization of specific reengineering tasks.

In 1995, Reasoning submitted a proposal to the Advanced Technology Program (ATP) to undertake technically high-risk research to develop software that would automate the reengineering and rewriting of legacy software systems. ATP awarded cost-shared funding to Reasoning through its focused program, Component-Based Software, which enabled the company to strategically position itself for the burgeoning Year 2000 (Y2K) repair market. Reasoning subsequently received more than $22 million in venture capital financing between 1996 and 2000 and grew from 12 people at the time of the ATP proposal in 1995 to more than 100 by 2000. At its peak, Reasoning made a significant impact as a leader in the Y2K software repair market, both as a seller of effective, low-cost software toolsets and as an innovator in software inspection tools. Today, after a sharp drop-off in the transformational software purchasing that had fueled its explosive pre-2000 growth, a leaner Reasoning has refocused its business model, secured an additional $9 million in venture funding, and continues to commercially market the software technology developed during the ATP project.

COMPOSITE PERFORMANCE SCORE

(based on a four star rating)

* *

Research and data for Status Report 94-06-0026 were collected during January 2002.

Legacy Systems Consume High Percentage of Corporate Resources

One of the key issues in significantly changing how software is composed and used is what to do with legacy systems. This is a problem not only for new software technologies, but also for day-to-day maintenance operations. In many large organizations, maintenance of legacy systems consumes more then 90 percent of information systems resources. A major difficulty is the wide variety of legacy systems, which might be written in any of dozens of computer languages and language "dialects," as well as countless specialized data formats. Companies that optimize business processes must often change legacy information systems to support the new processes. The required changes can involve new features, porting, performance optimization, or bug fixes. Minor changes can often be accomplished in a relatively painless fashion by modifying a small amount of code. However, major changes—such as porting a COBOL mainframe-based system to a UNIX client/server-based architecture built on a relational database—are typically very difficult, expensive, and risky.

Major changes often require a switch not only of code, but also of supporting tools (e.g., compilers and editors), development processes (testing and version control), and personnel. A major change is usually made by some combination of discarding part or all of the existing system, modifying existing parts, writing new parts, and purchasing new or improved parts from external vendors. If the change is accomplished primarily through discarding the existing system and buying or building new parts, the project is characterized as a rewrite or redevelopment. If the change is accomplished primarily by modifying the existing system, the project is characterized as a reengineering project. Rewriting and reengineering are the extremes along a spectrum of strategies for change; most major upgrades are accomplished by some combination of the two.

Reasoning offered software technology that would automate the rewriting and reengineering process for companies who needed to update legacy systems. However, to enter the larger software engineering market, it needed to make significant advances in its technological capabilities.

Expanding the Application of Data-Slicing Software

An organization's business policies, processes, and procedures are often maintained on its legacy systems, and it is inconceivable that these systems can simply be abandoned when a new system is purchased. It is equally implausible that all of an organization's legacy systems and applications can be rewritten or replaced as technology and business processes change. Therefore, maintaining, reengineering, and migrating these systems in a cost-effective and efficient manner is an important option.

In its 1995 ATP proposal, Reasoning proposed to apply the component-based automated code transformation technology that it had been refining since 1984 to the larger software reengineering industry. Reasoning proposed to use the principles of reusable software components and automated software composition to solve this pervasive problem by establishing the framework to easily create customized software reengineering tools. The company would create individual software components to handle interfaces with standard languages such as C, COBOL, and FORTRAN, and other components that implement sophisticated reengineering techniques, such as program data slicing, to build semantic models of the legacy system. Other companies would then be able to use these components to generate specific solutions for their customers; one application, for example, might be a software tool that extracts the implicit "business rules" from existing systems with a particular COBOL dialect and database.

An organization's business policies, processes, and procedures are often maintained on its legacy systems, and it is inconceivable that these systems can simply be abandoned when a new system is purchased.

In order to achieve its goal, Reasoning needed a combination of static and dynamic semantic analysis of software. The highlights of Reasoning's technical approach were the following:

- Automatic composition of reengineering tool components to support customization to specific jobs and reuse across multiple languages
- Use of program slicing and path feasibility analysis to derive semantic models of legacy code, such as business rules and system invariants
- Animation of code execution in terms of derived semantic models
- Analysis of derived semantic models to support reengineering and composition of the underlying legacy systems

Reasoning Proposes to Develop Slicing and Data Flow

Program slicing has been recognized in computer science research as a powerful technique for understanding programs. This technique allows the dissection and analysis of a program based on data flow. Using the analysis, it is then possible to answer many questions about how a program works. A key problem with program slicing, however, has been performance because the data-flow analysis necessary for program slicing is computationally intensive. To overcome performance issues with program slicing, Reasoning took a creative approach. It made the data-flow analysis incremental, an approach contrary to typical algorithms at the time that effectively computed the entire data-dependency graph for a program. Reasoning hoped that its advanced research on program slicing and its application to transformation systems would be a key differentiator in the marketplace.

Applications of the ATP-Funded Technology Could Be Diverse

By creating reusable components and automated composition techniques for adapting reengineering capabilities to diverse languages and databases, Reasoning's technology could reduce the time, cost, and risk of reengineering across a wide variety of legacy systems (e.g., COBOL-based business applications and FORTRAN scientific applications). First, improved reengineering productivity would lead to increased reuse of legacy systems, fewer disastrous "big-bang" redevelopment projects, extraction and exposure of business logic from legacy systems, and increased allocation of information systems resources to develop new systems instead of maintaining legacy systems. Second, by enabling cost-effective migration of legacy systems to new information technologies, demand for those technologies would be increased.

By creating reusable components and automated composition techniques, Reasoning's technology could reduce the time, cost, and risk of reengineering across a wide variety of legacy systems.

This would spur growth in high-value industries such as personal computers, workstations, networks, databases, multimedia, and software development. This growth would lead to higher investment in research and development (R&D), which would yield technological advances in those areas. Third, this technology would enable cost-effective migration of scientific software to new computing technologies, such as massively parallel computers that increase the productivity of scientists in software-intensive fields, from weather simulation to molecular biology.

ATP Support Accelerates Technology Development

To achieve commercial viability, however, Reasoning needed an infusion of capital to complete the research quickly before its international competitors caught up and eliminated its competitive advantage. Because the high-risk nature of this R&D project discouraged venture capitalists, Reasoning proposed a $2 million, three-year project to ATP. Without ATP funding, Reasoning would have pursued many of the same technical objectives of the project, but at only 10 to 20 percent of the funding level provided by the ATP award. This lower funding level would have significantly extended the schedule for delivering practical products. Reasoning's proposed technology was promising and the commercial advantage of rapid R&D was clear. Moreover, future applications of the technology had the potential to produce broad-based economic benefits beyond its own market by improving end-user and programmer productivity, as well as reducing high hardware and software maintenance costs.

Reasoning Achieves Technical Success

Bringing control to reengineering and rewriting projects through automation was Reasoning's stated high-level goal for the ATP-funded project. Furthermore, the company sought to design and prototype a framework and a set of components to formally capture legacy software systems and to build software reengineering, reverse engineering, and migration applications.

Reasoning achieved its objectives. Perhaps the most important goal that the company attained was the ability to help programs meet quality and dependability requirements by identifying and repairing defects in legacy systems. The company also built a framework that could easily be adapted to nonstandard languages and operating systems. No two legacy software systems are alike, and the problems that these systems manifest are diverse. Therefore, in order to successfully address legacy software, a toolset must be very flexible so that it can adapt to a variety of situations. Several of the key components that were developed demonstrated this power of flexibility.

ATP-Funded Technology Helps To Solve Y2K Problem

In 1996, companies were just beginning to address the Y2K problem. Reasoning's management identified this niche as a potential market for its ATP-funded technology, and, in 1999, began to target this profitable business. Reasoning attracted some top talent to drive this change in strategy and soon obtained venture capital support to begin commercializing its core technology to analyze, identify, and repair Y2K bugs in corporate computer systems. Analysts predicted that this market could be $300 to $900 billion in the years preceding 2000. Reasoning's growth was rapid. In three years, the company raised almost $23 million in venture capital investment, increased its staff from 12 to more than 100, and established a national presence with offices across the country. Reasoning successfully developed and marketed its Y2K solution and made a significant impact as a leader in the Y2K software repair market both as a seller of effective, low-cost software toolsets and as an innovator in software inspection tools. Reasoning's unique approach to solving the Y2K problem was based on the R&D conducted during the ATP project.

Investors Continue To Commit Funds to Inspection Tool

Reasoning transitioned its original software tool from a transformation tool to a Y2K tool, and, finally, to an inspection tool. ATP's funding support during the technology's critical years helped Reasoning create an innovative automated software inspection service that enables major technology companies to dramatically reduce the time, effort, and cost required to produce quality software.

Reasoning's unique approach to solving the Y2K problem was based on the R&D conducted during the ATP project.

Reasoning's solutions analyze the source code and pinpoint the exact location of crash-causing and data-corrupting defects before testing. With Reasoning's inspection database, a company can benchmark software quality across projects, companies, and industries.

Despite a reduction in force since 2000, Reasoning has reinvented itself and its technology several times in order to anticipate and react to changing market needs and conditions. A venture capital round of investment of $9.2 million in 2001 reaffirmed investors' commitment to Reasoning's technology. Reasoning is still developing the inspection tool with a restructured workforce of approximately 20 persons. The company remains hopeful that spending in the market will recover to pre-September 2001 spending levels.

Conclusion

What began as a highly academic and technical company became a fast-growing software firm that attracted top venture capital investors and recruited a proven management team from larger, public companies. ATP funding was the catalyst for Reasoning's new R&D capabilities and enabled it to become a viable software firm worthy of investment. As of January 2002, Reasoning continues to market the technology developed during the ATP project.

PROJECT HIGHLIGHTS
Reasoning Systems, Inc.

Project Title: Reengineering and Rewriting Legacy Software Systems (Component-Based Reengineering Technology)

Project: To use the principles of reusable software components and automated software composition to tackle the problem of numerous legacy systems within a corporation by establishing the framework for easily creating customized software reengineering tools.

Duration: 1/1/1995-12/31/1997
ATP Number: 94-06-0026

Funding (in thousands):

ATP Final Cost	$ 2,000	58%
Participant Final Cost	1,443	42%
Total	$ 3,443	

Accomplishments: Reasoning achieved the following goals during the ATP-funded project:

- Reduced the time, cost, and risk of reengineering across a wide variety of legacy systems by creating reusable components and automated composition techniques for adapting reengineering capabilities to diverse languages and databases

- Enabled cost-effective migration of scientific software to new computing platforms, such as massively parallel computers that increase the productivity of scientists in software-intensive fields, from weather simulation to molecular biology

Commercialization Status: Reasoning applied its ATP-funded technology to the Y2K problem. Since then, the company has marketed the technology developed during the ATP project. After being re-branded several times, the new technology provides automated software inspection services that enable major technology companies to dramatically reduce the time, effort, and cost required to produce quality software.

Outlook: Reasoning transitioned its original software tool from a transformation tool to a Y2K tool and, finally, to an inspection tool. The company has reinvented itself and its technology several times to anticipate and react to changing market needs and conditions. After receiving $23 million in venture capital investment between 1996 and 2000, a venture capital round of investment of $9.2 million in 2001 reaffirms the investment community's commitment to this technology as Reasoning refocuses on a new market.

Composite Performance Score: * *

Number of Employees: 12 employees at project start, 20 as of January 2002

Focused Program: Component-Based Software, 1994

Company:
Reasoning Systems, Inc.
700 East El Camino Real
Suite 300
Mountain View, CA 94040

Contact: Karl Schimpf
Phone: (650) 429-0350

Research and data for Status Report 94-06-0026 were collected during January 2002.